REMARKS:

In Office action mailed on October 2,2001, the examiner rejected claims 1-68 under 35 U.S.C.102(b) as anticipated by Okuzumi et al. The examiner also rejected claims 1-68 under 35 U.S.C. 102(b) as anticipated by Kenji et al.

By this Amendment, Applicant clarified independent claims 1, 5, 8, 12, 14, 18, 21, and 25. In making this revisions, care has been taken to ensure that the claims remain supported by the specification and that no new matter has been added.

Applicant appreciates the time and consideration provided by Examiner in reviewing this application, however, respectfully traverses the rejection of the claims at least for the following reasons.

Rejection under 35 U.S.C. 102(b)

Anticipation under 35 U.S.C. 102 requires that each and every claimed feature be disclosed by a single prior art reference. Therefore, the prior art reference relied upon by the Examiner in rejecting claims 1-68 must disclose an article that is reasonably identical to and includes at least every material element of the claimed method. Applicant respectfully submits that Okuzumi et al. and Kenji et al. do not disclose, let alone suggest, each and every claimed feature of rejected claims 1-68 in the cited references.

Independent Claims 1,5,8,12,14,18,21 and 25 were amended to recite that the respective claims define *executable* programs. Support for this amendment is found, for example, on page 4, lines 16, 24, the exemplary program that is provided in page 2, line 2 (Office 97 package). The detailed description (with reference to Figs. 2A-2B of the present application) describes a machine-instruction like structure.

A

REJECTION OVER OKUZUMI ET AL. (JP404242829A)

Before turning to discuss the Examiner's rejection on the merits, Applicants believe that describing certain aspects of the present application would assist in understanding the distinguishing features of the present invention over Okuzumi. Since the Examiner's detailed reasoning refers to Claim 1, the following arguments focus on Claim 1, and will later apply to the other Claims, *mutatis mutandis*.

The present application is aimed to improve the results obtained by difference extraction utilities, already known in the art, applied to executable programs (in the case of amended Claim 1) and generate a better (compact) result. A typical (yet not exclusive) scenario is when clients have an old executable program (say Windows 97 version) and the manufacturer would like to distribute updates of the Windows 97 (new executable program) to be reflected in the clients version. Applying a standard diff utility (between the old and new programs) would result in large diff file, which pose undue overhead insofar as transmission rate and reliability of transmission and update (at client side) are concerned. Applying the technique of the present application would result in compact diff file that can be easily transmitted and readily reflected in the clients' version.

General diff extraction utilities (see page 2, lines 14 and 15) were originally designed for extracting differences between two versions of a source program (referred to also as 'program source' or as 'source'). Sources are the original text files that programmers write in some formal language known as 'programming language'. Such sources are purely symbolic in the sense that they do not mention actual physical location of other elements in the source nor their absolute sequential location. Rather, any of required references in a source are made through symbolic names which themselves are part of the source.

Such diff utilities and methods are known in the art and known to be sufficient for extracting differences of such source

files. A major problem arises when applying these methods to executable program files aiming to extract the difference at the byte-level, regarding the files as a list of data bytes rather then list of text lines. The problem arises from the fact that executable programs are generated from sources and in that process many references are inserted into these executable files. These references do not refer symbolically to other location of the program, as may be the case in source files, but they refer to addresses - sequential locations in the program file. When re-generating an executable file from a modified source file, not only the actual changes are being reflected in the executable file, also the majority if not all the inserted references are modified as well since their referred addresses have changed their location in the executable file as a result of the actual changes. This phenomenon leads to a significant increase in the amount of differences. This problem is discussed in the "Background of the Invention" section of the specification, Page 2, lines 11 to 18, and exemplified on Page 2, lines 18 to 30.

Consider, for example, an extreme example where a change in the first source of line may lead to actual change of some first executable file, in which few bytes were added but also all references must change since they refer to locations that now have been moved farther for the amount of bytes added at the beginning. To simply reflect all the changed references when computing a difference, one must include them all. In accordance with the present application such a need is reduced or eliminated, and what is required, is just to send the first few modified bytes while computing the modified references at the time of reflecting the update on another old copy. Thus, in accordance with typical embodiments of the present application, a pre-processing is applied to both old and new files (giving rise to modified old and modified new programs) and applying a diff extraction utility to the modified old and modified new programs. The modification is effected in such a way that references become 'invariant' (see page 5, lines 22 to 29), thereby reducing the diff result and rendering it compact.

A

Okuzumi et al. explicitly mention 'source', and even more, this prior art reference contains a step of sorting 'statements' (a common name for an element of a source program) according to their 'character strings', such as in 0011.

In contrast, the present invention, according to amended Claim 1, defines *an executable* program. Accordingly, obtaining a compact difference result in the manner defined in the claims of the present application is not suggested even remotely in Okuzumi, which is not surprising considering the fact that in source programs there is no need to generate a compact result since the difference result is a priori compact.

In extracting diff between 2 versions of executable files as defined in amended Claim 1, there is no source involved, and neither statements, nor any textual or other symbolic representation of the program even exist.

Another major characteristic disclosed by Okuzumi et al. is that it deals with a special problem of updating an old source program (Okuzumi 0001, 0002 & 0008 & purpose section) that is being modified differently for 2 different copies and one is required to reflect both charges. For example, (see Okuzumi 00003) there is a package program (source program) that has undergone version change (first modified source). The same package program has been independently customized for a client use (second modified source). The version change of the first modified source need to be reflected also in the customized copy (second modified source). Note that the second modified source is not derived from the first modified source.

In contrast, in accordance with the present application the new program is derived from the old program (see for example page 18, lines 10 to 16, where P1 is the old program, P2 is the new program *"which was generated (or could have been generated) by the sequence of modifications as depicted in the imaginary memory table (80)). Said sequence of modifications (either real or imaginary), constitutes a transition sequence between* P_1 *and* P_2*"*.

A

This is an example of how P2 is derived from P1. There is a need to perform a "compact" diff between P1 and P2 and to this end modified P1 and modified P2 are generated.

Thus, not only the present invention as defined in amended Claims 1 concern executable program as compared to source programs in Okuzumi (and accordingly the need to obtain compact diff result in Okuzumi is obviated), but also the invention cannot be applied to a new program that is not derived from an old program, as is the case with Okuzumi.

Moreover, Claim 1 defines "preparatory" actions in connection with the references in order to produce the modified program (see Claim 1, steps a(i) and b(i)). Only after this preparatory action the diff operation is performed. In Okuzumi et al., there is no suggestion to apply preparatory actions before applying the diff and, a fortiori, not preparatory actions that pertain to the references. References in accordance with the present application are defined in page 4, lines 2 to 4, and are exemplified, for example, on page 18, lines 10 to 12 and in associated Fig. 2A. Note the references 5, 8, 1, 1, 13 and 11 (designated 41' through 46') as distinguished from addresses (1 to 15). The processing of the references and the reference entries is not even remotely suggested in Okuzumi and *a fortiori* not as a preparatory step for the application of diff. Note that by Okuzumi, "order table" is a table reflecting a specific re-ordering of the source statements. The latter not only directs to source and not to executable program, but also may only correspond to addresses and not to references.

Accordingly, Applicant respectfully submits that independent claims 1, 5, 8, 12, 14, 18, 21, and 25 as amended by the present amendment, are novel and allowable over the cited Okuzumi et al.

Dependent claims:

Claims 2,3, and 4.

Claims 2-4 are dependent directly or indirectly on

A

Claim 1, and therefore are also not anticipated by Okuzumi.

Claims 5-7,8-10,14-16,18-20,21-23,39-41,42-44,48-50,52-54,55-57

Claim 5 defines update of the receiving side using compact diff result from claim 1 and concerns executable programs, applying processing steps to the references (steps b(i) and (c)), which are not even remotely suggested by Okuzumi et al. Moreover, unlike Okuzumi, claim 5 concerns old and new programs which are derived one from the other. Applicant therefore submits that for the above reasons (some discussed in more detail with reference to Claim 1 above), Claim 5 is not anticipated by Okuzumi.

Claims 6,7 are dependent directly or indirectly on Claim 5, and therefore are also not anticipated by Okuzumi.

Claim 8 defines how to generate the compact difference result. It concerns executable programs, applying processing steps to the references steps b(i)), which characteristic is not even remotely suggested by Okuzumi et al. Moreover, unlike Okuzumi, claim 8 concerns old and new programs which are derived one from the other. Applicant therefore submits that Claim 8 is not anticipated by Okuzumi.

Claims 9,10 are dependent directly or indirectly on Claim 8, and therefore are also not anticipated by Okuzumi.

Claim 14 is a system claim that is not anticipated by Okuzumi for the reasons elaborated with reference to Claim 1.

Claims 15,16 are dependent directly or indirectly on Claim 14, and therefore are also not anticipated by Okuzumi.

Claim 18 is a system claim that is not anticipated by Okuzumi for the reasons elaborated with reference to Claim 5.

Claims 19, 20 are dependent directly or indirectly on Claim 18, and therefore are also not anticipated by Okuzumi.

Claim 21 is a system claim that is not anticipated

A

by Okuzumi for the reasons elaborated with reference to Claim 8.

Claims 22,23 are dependent directly or indirectly on Claim 21, and therefore are also not anticipated by Okuzumi.

Claims 35 to 68 are basically similar to claims 1 to 34, respectively, except for the fact that they recite data table instead of executable program. Data table is discussed on page 4, line 9 of the application and do not embrace source code as in Okuzumi. It is accordingly submitted that Claims 39-41, 42-44, 48-50, 52-54, 55-57 are not anticipated by Okuzumi for the reasons discussed in detail above with reference to Claims 1 to 3, 8-10, 14-16, 18-20, and 21-23.

Claims 11,13,17,24,26,28,31,34,38,45, 47,51,58,60,65 and 68.

It is also submitted that Claims 11,13,17,24,26,28,31, 34,38,45,47,51,58,60,65 and 68, are all dependent claims that recite storage medium, and not anticipated by Okuzumi for the reasons discussed in detail with reference to their corresponding independent claims.

Claims 12, 25, 35, 46, 59.

Wherein Claim 12 defines how to effect the update at the receiving side using a generated compact diff result. It concerns executable programs, applying processing steps to the references (steps b(i)), which are not even remotely suggested in Okuzumi. Moreover, unlike Okuzumi, claim 8 concerns old and new programs, which are derived one from the other. Applicant therefore submits that for reasons discussed in more detail with reference to Claim 8 and 1 above, Claim 12 is not anticipated by Okuzumi et al.

Claim 25 is similar to Claim 12 but is directed to a system and is not anticipated by Okuzumi et al. for the reasons elaborated with reference to Claim 12.

Claims 35,46 and 59 correspond to Clams 1, 12 and 25 except for the recitation of data table. Accordingly,

A

Applicant submits that for the reasons elaborated in connection with Claims 1, 12 and 25, the above claims are not anticipated by Okuzumi.

Claims 27, 61 and 62.

These dependent claims recite a storage medium and are not anticipated by Okuzumi for the reasons discussed in detail with reference to their corresponding independent Claims 1, 35 and 42.

Claims 29,32,36,63 and 66.

These dependent claims are not anticipated by Okuzumi, for the reasons elaborated in connection with Claim 2 above and their respective independent claims.

Claims 30, 33, 37, 64 and 67.

These dependent Claims are directed to the Internet and are not anticipated by Okuzumi et al. for the reasons discussed in detail with reference to their corresponding independent claims.

It is therefore submitted that all claims pending in the present application are novel and allowable over Okuzumi et al.

REJECTION OVER KENJI ET AL. (JP 05091550A)

In section 3, the Examiner rejected Claims 1 to 68 under 35 U.S.C. 102(b) as anticipated by Kenji et al. An English translation of Kenji (except for the "purpose" and "constitution" sections) was provided.

Before turning to discuss the rejection on the merits, a brief overview of Kenji et al. is provided for a better understanding of the distinguishing features between the present application and the teachings of Kenji. Kenji motivation and explicit wording is to minimize the time it takes to update a program *already* loaded in RAM using minimal time (Technical field of the invention - 0001). According to Kenji, some data processing devices have their program already loaded in RAM and the problem solved by Kenji et al. is to reduce the update of such devices from sending the whole

A

program to be loaded to just sending the differences (see operation Kenji 0006) with the addition that handles modules that expand their original size. In other words, there is an empty space *a priori* left between modules in the RAM and accordingly, if the "updated" (new) module is larger in size than the module that currently resides in the RAM (old module), the new module can fit the memory without affecting other modules (Kenji 0009). According to Kenji et al., the comparison is done " by address unit" (Kenji 0007), which means a simple scan on both memory images is applied, and matching addresses are each compared for finding the difference. Using such practice, even if 'references' would not exist at all in the program, insertion or deletion of even 1 single byte at the beginning of the program would cause the whole "program in the memory" to look different since the rest of the contents would shift and thus not match corresponding "addresses". Kenji et al. solve it by using "empty memory between modules" that in fact "absorbs" such shifts. So, if in the previous example, after 10 bytes there is an extra space of 1 byte, the amount of changes will not increase over 10 bytes (Delete is even simpler, since it does not require such absorption zones, it in fact creates them). Such a solution suggests some degree of compactness but based on an assumption of "empty" memory areas. It of course does not treat the issue of references at all.

Bearing this in mind, the distinguishing features between the claimed invention and Kenji et al. are as follow.

The present invention solves a different problem in a different manner. Thus, Kenji et al. deal with updating program in RAM, which is not the objective of the present application. The present invention aims at obtaining compact diff by applying a pre-processing step before applying the diff operation (see for exmple Claim 1, steps (a) and (b) that precede step (c)).

In Kenji et al., the "compactness" is based on the a

A

priori available space between the modules. The issue of leaving space between modules (in Kenji) in order to enable accommodation of updated modules is completely irrelevant for the present invention.

In the present application, the pre-processing is applied *inter alia* to the references. Kenji et al. do not mention even remotely references but only addresses. The distinction between address and reference is clear. Thus, for example, in Fig. 2A of the present application the references 5,8,1,1,13 and 11 (designated 41' to 46') are shown as distinguished from addresses 1 to 15. The processing of the references and the reference entries as claimed by the present application(in order to obtain compact diff), is not even remotely suggested in Kenji.

It is therefore submitted that Claim 1 is not anticipated by Kenji et al.

Claims 2, 3, and 4:

Claims 2-4 are dependent directly or indirectly on Claim 1, and therefore are also not anticipated by Kenji et al.

Claims 5-7,8-10,14-16,18-20,21-23,39-41,42-44,48-50,52-54,55-57:

The detailed distinctions between Kenji and the invention as claimed in Claim 1 also apply to claims 5-7,8-10,14-16,18-20,21-23,39-41,42-44,48-50,52-54,55-57 *mutatis mutandis*. It is therefore submitted that claims 5-7,8-10,14-16,18-20,21-23,39-41,42-44,48-50,52-54,55-57 are not anticipated by Kenji et al.

Claims 11,13,17,24,26,28,31,34,38,45,47,51,58,60,65 and 68.

The detailed distinctions between Kenji and the invention with reference to Claim 1 also apply to claims 11, 13, 17, 24, 26, 28,31, 34, 38, 45, 47, 51, 58, 60, 65 and 68 *mutatis mutandis*. It is therefore submitted that claims 11,

A

13, 17, 24, 26, 28, 31, 34, 38,45,47,51,58,60,65 and 68 are not anticipated by Kenji et al.

Claims 12,25,35,46,59.

The detailed distinctions between Kenji and the invention as claimed in Claim 1, also apply to claims 12,25,35,46,59 *mutatis mutandis*. Note that the nature of these claims is discussed in more detail with reference to Okuzumi above. It is therefore submitted that claims 12,25,35,46,59 are not anticipated by Kenji et al.

Claims 27, 61 and 62.

These dependent claims recite a storage medium and are not anticipated by Kenji for the reasons discussed in detail with reference to their corresponding independent Claims 1, 35 and 42.

Claims 29,32,36,63 and 66.

These dependent claims are not anticipated by Kenji et al. for the reasons elaborated in connection with Claim 2 above and their respective independent claims.

Claims 30, 33,37,64 and 67.

These dependent claims are directed to the Internet and are not anticipated by Kenji et al. for the reasons discussed in detail with reference to their corresponding independent claims.

Claims 27, 61 and 62.

These dependent claims recite a storage medium and are not anticipated by Kenji et al. for the reasons discussed in detail with reference to their corresponding independent Claims 1, 35 and 42.

Claims 29,32,36,63 and 66.

These dependent claims are not anticipated by Kenji et al. for the reasons elaborated in connection with Claim 2 above and their respective independent claims.

A

Claims 30, 33,37,64 and 67.

These dependent claims recite the Internet and are not anticipated by Kenji et al. for the reasons discussed in detail with reference to their corresponding independent claims.

Therefore, Applicant respectfully submits that the pending claims as amended by this Amendment are allowable over the cited prior art and the application is in condition for allowance.

The Commissioner is hereby authorized to charge any additional fees which may be required in this application under 37 C.F.R. §§1.16-1.17 during its entire pendency, or credit any overpayment, to Deposit Account No. 06-1135. Should no proper payment be enclosed herewith, as by a check being in the wrong amount, unsigned, post-dated, otherwise improper or informal or even entirely missing, the Commissioner is authorized to charge the unpaid amount to Deposit Account No. 06-1135. This sheet is filed in triplicate.

Respectfully submitted,

Kenneth H. Samples
Registration No. 25,747

Date: May 8, 2002

FITCH, EVEN, TABIN & FLANNERY
Suite 1600
120 South LaSalle Street
Chicago, Illinois 60603-3406
Telephone: (312) 577-7000
Facsimile: (312) 577-7007

A